小马哥的 Java 项目实战营

Java EE 项目 - 第五节 配置管理

小马哥（mercyblitz）

我是谁？

小马哥（mercyblitz）

- 父亲
- Java 劝退师
- Apache Dubbo PMC
- Spring Cloud Alibaba 架构师
- 《Spring Boot 编程思想》作者

议题

- 配置基础
- Apache Commons Configuration
- 问答互动

配置基础

- 简介

配置（Configuration）是应用程序重要的元信息（Metadata），几乎所有软件依赖配置调控程序行为，比如 MySQL 中的 my.ini 文件，Apache Maven 的配置文件 settings.xml 文件，以及其项目管理文件 pom.xml。尽管配置常以文件的形式承载，然而并仅限于此。根据配置所处的物理位置，可分为本地配置（Local Configuration）和远程配置（Remote Configuration）。其中，本地配置相对于程序的位置主要包括“内部配置（Internal Configuration）”以及“外部配置（External Configuration）”，而远程配置则有“版本化配置（Versioned Configuration）”和“分布式配置（Distributed Configuration）”等实现。

配置基础

- 本地配置（Local Configuration）

顾名思义，本地配置即配置存放在应用程序所在的物理环境，比如程序内部、物理机器或虚拟化容器。存放在程序进程内部的配置称之为“内部配置（Internal Configuration）”，相反则是“外部配置（External Configuration）”。

配置基础

- 内部配置（Internal Configuration）

通常，内部配置是通过程序代码，甚至是硬编码（Hard Code）实现。虽然不推荐程序使用硬编码来初始化配置，然而这并非是不可原谅的做法，如：

```
public Customizer<SentinelCircuitBreakerFactory> defaultConfig() {
    return factory -> {
        factory.configureDefault(
            id -> new SentinelConfigBuilder().resourceName(id)
            .rules(Collections.singletonList(new DegradeRule(id)
                                             .setGrade(RuleConstant.DEGRADE_GRADE_RT)
                                             .setCount(100)
                                             .setTimeWindow(10)))
            .build());
    };
}
```

配置基础

- 内部配置（Internal Configuration）

内部配置并未与硬编码（Hard-Code）划上等号，如 Java 8 引入的共享线程池 java.util.concurrent.ForkJoinPool#commonPool()：

```
private static ForkJoinPool makeCommonPool() {

    final ForkJoinWorkerThreadFactory commonPoolForkJoinWorkerThreadFactory =
        new CommonPoolForkJoinWorkerThreadFactory();
    int parallelism = -1;
    ......
        if (parallelism < 0 && // default 1 less than #cores
            (parallelism = Runtime.getRuntime().availableProcessors() - 1) <= 0)
            parallelism = 1;
    if (parallelism > MAX_CAP)
        parallelism = MAX_CAP;
    return new ForkJoinPool(parallelism, factory, handler, LIFO_QUEUE,
                            "ForkJoinPool.commonPool-worker-");
}
```

配置基础

- 外部配置（External Configuration）

外部配置是软件使用者、开发者以及运维者最常见和熟悉的配置手段，正如前文提到的 MySQL my.ini 文件等。在 Java 生态体系中，JDK 层面允许程序读取来自 Java System Properties、操作系统环境变量以及 JNDI 等多方配置来源。

```
private static ForkJoinPool makeCommonPool() {

    final ForkJoinWorkerThreadFactory commonPoolForkJoinWorkerThreadFactory =
            new CommonPoolForkJoinWorkerThreadFactory();
    int parallelism = -1;
    try {  // ignore exceptions in accessing/parsing properties
        String pp = System.getProperty
            ("java.util.concurrent.ForkJoinPool.common.parallelism");
        ......
        if (pp != null)
            parallelism = Integer.parseInt(pp);
```

配置基础

- 远程配置（Remote Configuration）

在分布式场景中，远程配置的内容来自于配置运用端程序进程物理环境以外的环境，其作用相当重要，是大多数互联网企业的基础设施标配，大多数实现采用 C/S（客户端/服务器）架构。C/S 架构广义地包含了 B/S（浏览器/服务器）架构，这两种实现均采用 B/S 架构，属于“分布式配置（Distributed Configuration）B/S 架构，然而都归类于“分布式配置实现，即应用需要依赖一个专属的 HTTP 客户端，通过访问远端的 Web 服务器获取配置。实现上，配置客户端版本化配置（Versioned Configuration）”。

配置基础

- Java 标准外部化配置
  - Java SE
    - Java 系统属性 - java.lang.System#getProperties()
    - 操作系统环境变量 - java.lang.System#getenv()
    - 偏好配置 - java.util.prefs.Preferences
  - Java EE
    - Servlet 上下文配置 - javax.servlet.ServletContext#getInitParameter(String)
    - Servlet 配置 - javax.servlet.ServletConfig#getInitParameter(String)
    - JSP 配置 - javax.servlet.descriptor.JspConfigDescriptor
    - JNDI 配置 - javax.naming.Context#lookup(javax.naming.Name)

Apache Commons Configuration

- Apache 通用配置开源框架
  - 支持数据源
    - Properties files
    - XML documents
    - Windows INI files
    - Property list files (plist)
    - JNDI
    - JDBC Datasource
    - System properties
    - Applet parameters
    - Servlet parameters

Apache Commons Configuration

- 配置源
  - 接口信息
    - PropertiesConfiguration Loads configuration values from a properties file
    - XMLConfiguration Takes values from an XML document
    - INIConfiguration Loads the values from a .ini file as used by Windows
    - PropertyListConfiguration Loads values from an OpenStep .plist file
    - JNDIConfiguration Using a key in the JNDI tree, can retrieve values
    - BaseConfiguration An in-memory method of populating a Configuration object
    - HierarchicalConfiguration An in-memory Configuration object
    - SystemConfiguration A configuration using the system properties

配置管理

- 数据类型
  - BigDecimal
  - BigInteger
  - boolean
  - byte
  - double
  - float
  - int
  - long
  - short
  - String

配置管理

- 操作方法
  - addProperty()
    - Adds a new property to the configuration. If this property already exists, another value is added to it (so it becomes a multi-valued property)
  - clearProperty()
    - Removes the specified property from the configuration
  - setProperty()
    - Overwrites the value of the specified property. This is the same as removing the property and then calling addProperty() with the new property value.
  - clear()
    - Wipes out the whole configuration

Apache Commons Configuration

- Apache Commons Configuration 1.x - 使用
  - Maven 坐标

```
<dependency>
    <groupId>commons-configuration</groupId>
    <artifactId>commons-configuration</artifactId>
    <version>1.10</version>
</dependency>
```

  - 依赖
    - http://commons.apache.org/proper/commons-configuration/dependencies_1_10.html [点击访问]

THANKS!